Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1 (Original). A system for the concurrent operation of plural computer applications, said system comprising:

(a) a shared object space capable of simultaneous connection with plural said applications and capable of storing at least one object accessible to at least two said applications when said plural applications are connected with said shared object space;

(b) a lock table for storing lock nodes, each said lock node being uniquely associated with a one of said plurality of applications; and

(c) said at least one object having an object header capable of storing any selective one of:

(1) a selective said lock node;

(2) a reference to a selective one of said lock nodes stored in said lock table; and

(3) a default value other than one of said lock nodes or a reference to one of said lock nodes stored in said lock table.

2 (Original). The system of claim 1 wherein said default value is zero.

3 (Original). The system of claim 1 where said lock table is located in said shared object space.

4 (Original). The system of claim 1 where said selective one of said lock nodes operates as a cheap lock when stored in said object header.

5 (Original). The system of claim 4 where said reference to a selective one of said lock nodes stored in said lock table operates as an expensive lock when stored in said object header.

6 (Original). The system of claim 1 where said reference to a selective one of said lock nodes stored in said lock table operates as an expensive lock when stored in said object header.

7 (Original). The system of claim 6 where a said application that has a said expensive lock to a said at least one object grants said lock to another said application by swapping one of said selective one of said lock nodes, said reference, and said default value for another one of said selective one of said lock nodes, said reference, and said default value.

8 (Original). The system of claim 1 including a lock manager that controls access to at least one object by selectively swapping one of said selective one of said lock nodes, said reference, and said default value for another one of said selective one of said lock nodes, said reference, and said default value.

9 (Original). The system of claim 8 where said swap is atomic.

10 (Original). The system of claim 1 where said at least one object comprises a plurality of sub-objects, each said sub-object having a said object header.

11 (Original). The system of claim 1 where each said plural application runs in its own virtual machine.

12 (Original). The system of claim 1 each said virtual machine is a Java virtual machine.

13 (Currently amended). The system of claim 12 where said shared object space is connected to each said Java virtual machine by a ~~Java~~ native method interface.

14 (Original). The system of claim 1 where at least one of said plural applications is not an object oriented application.

15 (Original). The system of claim 14 where said at least one of said plural applications is a C application.

16 (Original). A method for synchronizing access to a shared object stored in a shared object space by plural computer applications, said method comprising:

(a) storing at least one object in a shared object space so that said object is accessible to at least two of said plural applications when said plural applications are connected to said shared object space, said at least one object having an object header;

(b) selectively storing at least one lock node in a lock table, each said lock node being uniquely associated with a one of said plurality of applications; and

(c) storing in said object header any selective one of;

(1) a selective said lock node;

(2) a reference to a selective one of said lock node stored in said lock table; and

(3) a default value other than one of said lock nodes or a reference to one of said lock nodes stored in said lock table.

17 (Currently amended). The method ~~system~~ of claim 16 where said default value is zero.

18 (Currently amended). The method ~~system~~ of claim 16 where said lock table is located in said shared object space.

19 (Currently amended). The method ~~system~~ of claim 16 where said selective one of said lock nodes operates as a cheap lock when stored in said object header.

20 (Currently amended). The method ~~system~~ of claim 19 where said reference to a selective one of said lock nodes stored in said lock table operate as an expensive lock when stored in said object header.

21 (Currently amended). The method ~~system~~ of claim 16 where said reference to a selective one of said lock nodes stored in said lock table operates as an expensive lock when stored in said object header.

22 (Currently amended). The method ~~system~~ of claim 21 where a said application that has a said expensive lock to a said at least one object grants said lock to another said application by swapping one of said selective one of said lock nodes, said reference, and said default value for another one of said selective one of said lock nodes, said reference, and said default value.

23 (Currently amended). The method ~~system~~ of claim 16 including a lock manager that controls access to at least one object by selectively swapping one of said selective one of said lock nodes, said reference, and said default value for another one of said selective one of said lock nodes, said reference, and said default value.

24 (Currently amended). The method ~~system~~ of claim 23 where said swap is atomic.

25 (Currently amended). The method ~~system~~ of claim 16 where said at least one object comprises a plurality of sub-objects, each said sub-object having a said object header.

26 (Currently amended). The method ~~system~~ of claim 16 where each said plural application runs in its own virtual machine.

27 (Currently amended). The method ~~system~~ of claim 26 where each said virtual machine is a Java virtual machine.

28 (Currently amended). The method ~~system~~ of claim 27 where said shared object space is connected to each said Java virtual machine by a ~~Java~~ native method interface.

29 (Currently amended). The method ~~system~~ of claim 16 where at least one of said plural applications is not an object oriented application.

30 (Currently amended). The method ~~system~~ of claim 29 where said at least one of said plural applications is a C application.

31 (Original). A system for the concurrent operation of plural computer applications, said system comprising:

(a) a shared object space capable of simultaneous connection with plural said applications and capable of storing at least one object accessible to at least two said applications when said plural applications are connected with said shared object space;

(b) said at least one object having a header capable of storing a lock that indicates that one of said plural applications is using the object associated with said header; and

(c) said lock being either a cheap lock or an expensive lock, said cheap lock defined as a lock that may be inserted into or removed from said header in a relatively short time with respect to said expensive lock.

32 (Original). The system of claim 31 where said header stores an expensive lock when one said application is waiting for said object associated with said header while another said application is using said object associated with said header, and said lock is a cheap lock otherwise.

33 (Original). The system of claim 31 including a lock table capable of storing a plurality of lock nodes, each said lock node being uniquely associated with a one of said plurality of applications.

34 (Original). The system of claim 33 where each said application is capable of using said at least one object through a plurality of threads and said lock table stores information as to which threads are waiting for an object being used.

35 (Original). The system of claim 34 where said threads waiting for said object being used are prioritized to determine which of said threads next gets access to said object being used.

36 (Original). The system of claim 34 where said threads waiting for said object being used are not prioritized to determine which of said threads next gets access to said object being used.

37 (Original). The system of claim 31 where said header contains a default value when said header does not contain a lock.

38 (Original). The system of claim 37 where said default value is zero.

39 (Original). The system of claim 33 where said lock table is located in said shared object space.

40 (Original). The system of claim 33 where a said application that has a said expensive lock to a said at least one object grants said lock to another said application.

41 (Original). The system of claim 33 including a lock manager that controls access to at least one object.

42 (Original). The system of claim 31 where said at least one object comprises a plurality of sub-objects, each said sub-object having a said object header.

43 (Original). The system of claim 31 where each said plural application runs in its own virtual machine.

44 (Original). The system of claim 43 where each said virtual machine is a Java virtual machine.

45 (Currently amended). The system of claim 44 where said shared object space is connected to each said Java virtual machine by a ~~Java~~ native method interface.

46 (Original). The system of claim 31 where at least one of said plural applications is not an object oriented application.

47 (Original). The system of claim 46 where said at least one of said one said plural applications is a C application.